Software Engineering Institute

Evaluation of the Applicability of HTML5 for Mobile Applications in Resource-Constrained Edge Environments

Bryan Yan
Grace A. Lewis

July 2014

TECHNICAL NOTE
CMU/SEI-2014-TN-002

Critical System Capabilities

http://www.sei.cmu.edu

Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

DM-0001051

Table of Contents

List of Figures

List of Tables

Abstract

Mobile applications increasingly are being used by first responders and soldiers to support their missions. These users operate in resource-constrained, edge environments characterized by dynamic context, limited computing resources, intermittent network connectivity, and high levels of stress. In addition to efficient battery management, mobile applications operating in edge environments require efficient resource usage of onboard sensors to capture, store, and send data across networks that may be intermittent. The traditional method for building mobile applications is to use native software development kits (SDKs) on a particular mobile platform, such as Android or iOS. However, HTML5 has recently evolved to a stage where it supports many of the development features that native SDKs support. The advantages of using HTML5 not only include cross-platform development and deployment, but also that mobile edge applications would not have to be deployed on mobile devices, potentially leading to an easier distribution and testing process because they simply run inside the web browser that already exists on the device. This technical note presents an analysis of the feasibility of using HTML5 for developing mobile edge applications, as well as the use of bridging frameworks for filling in gaps in HTML5 development features. This note also provides a discussion of the software architecture implications of HTML5 mobile application development. The work presented in this note is the result of an independent study in Carnegie Mellon University's Master of Information Technology - Embedded Software Engineering (MSIT-ESE) program.

1 Introduction

Mobile applications increasingly are being used by first responders and soldiers to support their missions. These users operate in resource-constrained, edge environments; not only are they at the edge of the network infrastructure, but they are also resource constrained due to dynamic context, limited computing resources, intermittent network connectivity, and high levels of stress.

Typical mobile applications used by field personnel include face recognition, speech recognition, natural language translation, and situational awareness. These types of applications require access to sensors on the mobile device to capture data, and using the network interfaces to send data to nearby servers or the cloud, if the data cannot be processed locally. The natural way to support this type of functionality is through native[1] mobile applications. However, native applications present certain disadvantages:

- They are tied to a particular mobile platform such as Android or iOS.
- They must be deployed onto the devices typically through an app store (application store of the mobile device manufacturers or internal to an organization from which apps can be downloaded).
- They require a complex testing process for each platform (many app stores have very strict testing processes that have to be passed before an application can be made available to users).

Alternatively, HTML5 is a technology that can be used to develop mobile applications [W3C 2013]. The advantages of using HTML5 include portability across mobile device platforms as well as mobile device capability to access functionality without having to download and install applications. However, it is not clear that HTML5 can support all the development features required by edge applications. This paper evaluates the use of HTML5 for developing edge applications and explores the use of alternative technologies such as bridging frameworks to meet edge application requirements.

Section 2 describes related work on using HTML5 for mobile application development. Section 3 provides background information on edge applications, HTML5 mobile applications, and hybrid mobile applications. Section 4 presents the initial evaluation of building edge applications using HTML5. Section 4 presents the use of bridging frameworks to satisfy some of the application development features required by edge applications that are not met by using HTML5 alone. Section 5 describes some of the software architecture implications of developing edge applications using HTML5. Section 6 concludes and summarizes the work.

[1] Native mobile applications are developed, tested, and deployed for a specific mobile application platform (e.g., Android, iOS, Blackberry, Windows Phone) and installed directly onto the device itself.

2 Background

2.1 Edge Applications

We define edge applications as mobile applications operating at the edge of the network infrastructure, to support first responders and military personnel in their execution of tasks and missions. These edge environments are characterized by

- intermittent or no connectivity to the enterprise
- a fast-paced, highly fluid and unpredictable environment due to, for example, mission changes, threats, or changing weather condition
- the potential of involving large amounts of field-collected data
- resource challenges (power, computing, etc.)
- periods of very high stress and cognitive load

To deal with these characteristics, edge applications must

- exploit available sensors such that contextual information can be captured with easy and minimal user interaction
- process, store, and forward sensed and captured information
- be resilient to intermittent communications connectivity and opportunistic in using communication capabilities as they become available
- manage resources on the mobile device such that they are used as efficiently as possible to maximize the availability of the system by reducing power consumption

Table 1 lists some of the mobile application development features for edge applications that are required to satisfy the previous list of requirements, along with rationale for the features.

Table 1: Edge Application Development Features

Feature	Rationale
geolocation	display of user location and location of relevant events
Wi-Fi communication – client-server	opportunistic communication with servers
Wi-Fi communication – peer-to-peer	opportunistic communication with peers and nearby devices
Bluetooth communication	opportunistic communication with peers and nearby servers and devices
background processing	background services for sensor gathering and communications
interactive user interface (such as touch interface and gestures)	easy interaction with device in stressful conditions
user notification	alerting of events
access to files	access to configuration files, user preferences, and sensed data

accelerometer	activity recognition
orientation	map panning based on the direction that the user is facing
microphone	data input, triangulating the source of a sound, push-to-talk communications
compass	map panning based on the direction the user is facing
ambient light	activity recognition, sensing of environmental conditions
proximity	activity recognition, sensing of environmental conditions
ambient temperature	activity recognition, sensing of environmental conditions
ambient humidity	activity recognition, sensing of environmental conditions
atmospheric pressure	activity recognition, sensing of environmental conditions
battery status	mobile device battery life optimization such as reducing low priority processes under low battery conditions
audio playback	receiving information audibly
video playback	receiving information visually
camera	capturing surrounding environment visually
vibration	alerting of events

2.2 HTML5 Mobile Applications

HTML5 is the fifth revision of the HTML specification developed by the World Wide Web Consortium (W3C) [W3C 2013]. HTML5 mobile applications are similar to web applications in that the technologies used for application development are the same: HTML, JavaScript, and CSS3. To specifically address mobile devices, HTML5 APIs have been drafted by W3C for the use of on-board sensors such as the accelerometer [Block 2011].

The major benefits of using HTML5 for mobile application development are the portability across mobile platforms and an easier testing, deployment, and distribution process, because applications do not need to be installed on mobile devices. The distribution aspect, especially, can be challenging for distributed teams operating in edge environments. However, the major drawback is that the APIs to access the mobile devices' hardware and on-board sensors are limited for use in edge applications, as discussed in Section 4 of this paper.

2.3 Web Application Bridging Frameworks

Web application bridging frameworks are software packages created by third-party software development organizations that merge the advantages of native mobile application development with the advantages of HTML5. Examples of bridging frameworks include PhoneGap and SenchaTouch. The applications created by utilizing bridging frameworks are referred to as *hybrid mobile applications*. These applications have the advantage of high portability—because the framework can compile the hybrid mobile applications to multiple mobile platforms—as well as the ability to access on-board sensors of mobile devices.

Although developing hybrid mobile applications using bridging frameworks seems like a perfect fit for edge application development requirements, there are problems with performance of hybrid mobile applications. In addition, because the applications are compiled into a form of mobile app, the distribution problem of native mobile applications remains. The details of using bridging frameworks are discussed in Section 4.

3 Initial Evaluation: HTML5

To determine if edge applications could be developed using HTML5 technology, we started implementing all the development features listed in Table 1 and testing the code on the Google Chrome, Firefox for Android, and Dolphin web browsers, on a Google Nexus 7 tablet and a Samsung Galaxy S4 smartphone, running Version 4.3 and 4.2.1 of the Android OS respectively. Table 2 shows the list of edge application development features and their corresponding HTML5 support. "Fully supported" means that the feature can be fully developed using only HTML5. "Not currently supported" means that the feature is not currently supported by the HTML5 specification, but a draft HTML5 specification is forming with a claim that mobile web browsers will soon support the feature. "Not supported" means that HTML5 cannot be used to support the feature. The details of each edge application development feature and its corresponding HTML5 support are discussed in detail in the following subsections.

Table 2: HTML Support for Edge Applications

Edge Application Development Feature	**HTML5 Support as of December 2013**
geolocation	fully supported
Wi-Fi communication – client-server	fully supported
Wi-Fi communication – peer-to-peer	fully supported
Bluetooth communication	not supported
background processing	fully supported
interactive UI (such as touch interface and gestures)	fully supported
user notification	not currently supported
access to files	fully supported
accelerometer	fully supported
orientation	fully supported
microphone	fully supported
compass	fully supported
ambient light	not currently supported
proximity	not currently supported
ambient temperature	not currently supported
ambient humidity	not currently supported
atmospheric pressure	not currently supported
battery status	not currently supported
audio playback	fully supported
video playback	fully supported

camera	fully supported
vibration	not currently supported

3.1 Geolocation

The HTML5 standard has full geolocation support for mobile devices as shown in the code example in Figure 1. The example registers the location tracking feature of the mobile device and reports the longitude, latitude, altitude, heading, speed, and accuracy of the location in plain text format.

```
<!DOCTYPE html>
<html>
   <body>
      <div id="GeoDemo"></div>
      <script>
         var startTime = new Date().getTime();

         function success(pos) {
               var crd = pos.coords;
               var toPrint = "<p>Your current position is:</p>";
               toPrint += "<p>Latitude : " + crd.latitude + "</p>";
               toPrint += "<p>Longitude: " + crd.longitude + "</p>";
               toPrint += "<p>Altitude: " + crd.altitude + "</p>";
               toPrint += "<p>Heading: " + crd.heading + "</p>";
               toPrint += "<p>Speed: " + crd.speed + "</p>";
               toPrint += "<p>Accuracy within " + crd.accuracy + " meters.</p>";
               toPrint += "<p>Retrieved location in " + (new Date().getTime() - startTime) + "
                   milliseconds.</p>";
               document.getElementById("GeoDemo").innerHTML = toPrint;
               navigator.geolocation.clearWatch(loc_id);
         };

         function error(err) {
               console.warn('ERROR(' + err.code + '): ' + err.message);
               document.getElementById("GeoDemo").innerHTML = "<p>Cannot access device's
                   location.</p>";
               navigator.geolocation.clearWatch(loc_id);
         };

         var options = {
               enableHighAccuracy : true,
               timeout : 5000,
               maximumAge : 0
         };

         var loc_id = navigator.geolocation.watchPosition(success, error, options);

      </script>
   </body>
</html>
```

Figure 1: HTML5 Code Example for Geolocation

3.2 Wi-Fi Communication

There are two possible architectures for use of Wi-Fi communication to support edge applications: client-server and peer-to-peer. In the client-server architecture, the mobile device is acting as a client and using Wi-Fi to communicate with a server. In the peer-to-peer architecture the mobile device is using Wi-Fi to communicate with another mobile device.

- client-server architecture: fully supported
- peer-to-peer architecture: fully supported

Wi-Fi communication in the traditional client-server architecture is fully supported using the WebSockets interface. However, the precondition is that a computing machine such as a desktop computer or a server machine must be set up as a WebSockets server. For this reason, peer-to-peer Wi-Fi communication is supported but cannot be achieved through HTML5 WebSockets alone.

Figure 2 shows the code example that demonstrates the utilization of the HTML5 WebSockets API. The code example connects to a WebSockets server set up by "echo.websocket.org" and then sends the message "WebSockets Rocks" to that server. The server replies back with the same message.

To support peer-to-peer Wi-Fi communication on mobile devices using HTML5 via client-side scripting, an open source project called WebRTC enables web browsers to communicate to each other in real time via simple JavaScript [Google 2011]. Although the communication and data transmission that takes place between mobile devices is peer-to-peer, the initial connection setup or "signaling" stage requires connection to a server, such as a WebSockets server or a server that implements the Internet Communications Engine (ICE) framework [Dutton 2012].

WebRTC is a newly formed standard and is supported by Android Google Chrome version 29 as of August 21, 2013, according to The Chromium Blog [Chromium 2013], and by Firefox for Android as of September 17, 2013, according to The Mozilla Blog [Mozilla 2013a].

```
<!DOCTYPE html>
<meta charset="utf-8" />
<title>WebSocket Test</title>
<script language="javascript" type="text/javascript">
   var wsUri = "ws://echo.websocket.org/";
   var output;
   function init() {
        output = document.getElementById("output");
        testWebSocket();
   }

   function testWebSocket() {
        websocket = new WebSocket(wsUri);
        websocket.onopen = function(evt) {
                onOpen(evt)
        };
        websocket.onclose = function(evt) {
                onClose(evt)
        };
        websocket.onmessage = function(evt) {
                onMessage(evt)
        };
        websocket.onerror = function(evt) {
                onError(evt)
        };
   }

   function onOpen(evt) {
        writeToScreen("CONNECTED");
        doSend("WebSockets Rocks");
   }

   function onClose(evt) {
        writeToScreen("DISCONNECTED");
   }

   function onMessage(evt) {
        writeToScreen('<span style="color: blue;">RESPONSE: ' + evt.data + '</span>');
        websocket.close();
   }

   function onError(evt) {
        writeToScreen('<span style="color: red;">ERROR:</span> ' + evt.data);
   }

   function doSend(message) {
        writeToScreen("SENT: " + message);
        websocket.send(message);
   }

   function writeToScreen(message) {
        var pre = document.createElement("p");
        pre.style.wordWrap = "break-word";
        pre.innerHTML = message;
        output.appendChild(pre);
   }

   window.addEventListener("load", init, false);
</script>
<h2>WebSocket Test</h2>
<div id="output"></div>
</html>
```

Figure 2: HTML5 Code Example for Using WebSockets[2]

[2] ©2013 Kaazing Corporation. Reprinted with permission (*http://www.websocket.org/echo.html*).

3.3 Bluetooth Communication

Bluetooth communication is not supported. There are no Bluetooth specifications in the HTML5 draft, nor are there third-party client-side APIs that support Bluetooth communication in mobile devices.

3.4 Background Processing

Background processing occurs when a process can continue to execute even when it is not the user's currently active process. For example, if a user is running a timer process to track the time of an activity, and then switches to watch a video and hides the timer process (for example, using the Home button on an Android device), the timer process should continue to run and keep track of activity time if background processing is allowed.

Background processing of tasks is supported fully by HTML5 via the WebWorkers API specified by W3C [Hickson 2012]. The caveat is that the browser that is running the task still must be in running state within the Android OS. If the web browser process stops running, then the task that was sent to the background will no longer execute.

Figure 3 shows the HTML5 code example that demonstrates the background processing capability. The example uses a Worker object that runs in the background to count the seconds elapsed since the last time that the HTML5 web page was opened. Note that the Worker object must take an executable JavaScript file as a parameter. In the example, timer.js is passed into the Worker object, which is shown in Figure 4.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Background processing example</title>
   </head>
   <body>
      <p>
      <h1>Number of seconds since you opened this page:
         <output id="result">
         0
         </output>
      </h1>
      <br/>
      <br/>
      Notice that if you switch to another tab, or put Google Chrome to the background, the
      timer still runs. The Web Worker will stop only if the Google Chrome process is killed.
      </p>
      <script>
         var worker = new Worker('timer.js');
         worker.onmessage = function(event) {
               document.getElementById('result').textContent = event.data;
         };
      </script>
   </body>
</html>
```

Figure 3: HTML5 Code Example for Background Processing

```
var timer = 0;

setInterval
(
        function()
        {
                timer ++;
                postMessage(timer);
        },
        1000
);
```

Figure 4: Timer.js JavaScript File

3.5 Interactive UI

Interactive UI is fully supported by the HTML5 standard. W3C has drafted the Touch API for mobile devices to handle simple single-point and multi-point touches for complex gestures and gesture-based transformations [Schepers 2013].

Figure 5 shows the code example that demonstrates part of the HTML5 Touch API. The example lets the user drag a picture around the screen: as the user holds his or her finger down on the picture and moves it around, the picture will follow the tip of the finger until the user lets go.

```
<!DOCTYPE html>
<html>
   <head></head>
   <body onload="startup()">
      <h2>Android Users: Drag Me!</h2>
      <br \="">
      <div id="TouchDemo">
         <img style="border: 10px solid ; width: 150px; height: 200px;"
            src="Turnipe_Cake_TO.jpg" alt="">
      </div>
      <script>
         //global variables to keep track of the picture's location
         var el = document.getElementById("TouchDemo");
         var posX = 0, posY = 0, lastPosX = 0, lastPosY = 0;

         function startup() {
               el.addEventListener("touchstart", handleStart, false);
               el.addEventListener("touchend", handleEnd, false);
               el.addEventListener("touchcancel", handleCancel, false);
               el.addEventListener("touchleave", handleEnd, false);
               el.addEventListener("touchmove", handleMove, false);
         }

         function handleStart(evt) {
               evt.preventDefault();
               posX = evt.touches[0].pageX;
               posY = evt.touches[0].pageY;
         }

         function handleMove(evt) {
               evt.preventDefault();
               el = document.getElementById("TouchDemo");

               //the new positions of the image,
               //but needs to be offset by the height and width to center the drag
               posX = evt.touches[0].pageX + lastPosX - 90;
               posY = evt.touches[0].pageY + lastPosY - 150;

               if (posX < 0) {
                     posX = 0;
               }
               if (posY < 0) {
                     posY = 0;
               }
               //the major method for altering the image, there is also rotation and scale
               var transform = "translate3d(" + posX + "px," + posY + "px, 0)";
               el.style.webkitTransform = transform;
         }

         function handleEnd(evt) {
               evt.preventDefault();
               lastPosX = evt.touches[0].pageX;
               lastPosY = evt.touches[0].pageY;
         }

         function handleCancel(evt) {
               evt.preventDefault();
         }
      </script>
   </body>
</html>
```

Figure 5: HTML5 Code Example for Interactive UI

3.6 User Notification

User notification is defined as the alert texts, sounds, or device vibration that the user receives when an important event has occurred, such as when receiving a text message from another user even if the text messaging application is not open in the foreground.

The user notification feature is not currently supported by the Android mobile web browsers as of December 2013, even though there is an HTML5 notification API in the draft stage [Gregg 2013]. However, there is a third-party JavaScript API called Pusher that simulates the notification feature by overlaying an HTML block on top of the webpage when a notification is received. Developers can embed the Pusher API to send, receive, and display notifications in text form in real time [Pusher 2012]. The Pusher API is closed-source and therefore we cannot show a code example.

3.7 Access to Files

Access to files is fully supported by the HTML5 standard via the HTML5 File API drafted by W3C [Ranganathan 2013].

Figure 6 shows a JavaScript code example that demonstrates the use of this API. It enables the user to select a file and displays some of the file properties such as file size and last-modified date.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>File API Examples</title>
   </head>
   <h1>File API Example</h1>
   <br/>
   <div>
      Click on the button below to select a file to inspect
   </div>
   <input type="file" id="files" name="files[]" multiple />
   <output id="list"></output>
   <script>
      function handleFileSelect(evt) {
        var files = evt.target.files;
        // FileList object

        // files is a FileList of File objects. List some properties.
        var output = [];
        for (var i = 0, f; f = files[i]; i++) {
                output.push('<li><strong>', escape(f.name), '</strong> (', f.type || 'n/a', ')
                   - ', f.size, ' bytes, last modified: ', f.lastModifiedDate ?
                   f.lastModifiedDate.toLocaleDateString() : 'n/a', '</li>');
        }
        document.getElementById('list').innerHTML = '<ul>' + output.join('') + '</ul>';
      }
      document.getElementById('files').addEventListener('change', handleFileSelect, false);
   </script>
</html>
```

Figure 6: JavaScript Code Example for Accessing Files [3]

[3] Reprinted from Bidelman [Bidelman 2010]. Code examples are subject to the Apache 2.0 License (http://www.apache.org/licenses/LICENSE-2.0).

It is important to note that some mobile browsers, such as Google Chrome for Android, have restricted HTML5 file access to certain directories, while other mobile browsers, such as Firefox for Android and Dolphin, have access to more directories. For example, during test runs, Google Chrome was able to access photos only from the pictures directory, while the Firefox for Android and Dolphin web browsers were able to access files from more than just the pictures directory.

3.8 Accelerometer

The accelerometer feature is fully supported by the HTML5 standard device motion API drafted by W3C [Block 2011]. The test mobile web browsers on the Android platform have adopted this API.

Figure 7 shows the code example that demonstrates the use of this API. The code example accesses the accelerometer of the mobile device and displays the acceleration or delta movements of the mobile device in x, y, and z coordinates. The details of the operations can be found on the W3C website [Block 2011].

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Accelerometer</title>
   </head>
   <body>
      <p>
      <h1 id="accelerometer"></h1>
      </p>
      <script>
         window.addEventListener("devicemotion", function(event) {
               // Process event.acceleration
               var keepDigits = 1000000;
               document.getElementById("accelerometer").innerHTML = "X: " +
                    parseInt(event.acceleration["x"] * keepDigits) + "</br> Y: " +
                    parseInt(event.acceleration["y"] * keepDigits) + "</br> Z: " +
                    parseInt(event.acceleration["z"] * keepDigits);
         }, true);
      </script>
   </body>
</html>
```

Figure 7: HTML5 Code Example for Accelerometer

3.9 Orientation

Orientation is fully supported by the HTML5 standard via the device orientation API. This standard was drafted by W3C to support accessing the mobile device's ability to determine its own orientation: portrait or landscape. The test mobile web browsers on the Android platform have adopted this API.

Figure 8 shows the code example that demonstrates the use of this API. It accesses the mobile device's orientation, which is represented by the variables alpha, beta, and gamma. The details of the operations can be found on the W3C website [Block 2011].

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Orientation</title>
   </head>
   <body>
      <p>
      <h1 id="orientation"></h1>
      </p>
      <script>
         window.addEventListener("deviceorientation", function(event) {
               // process event.alpha, event.beta and event.gamma
               document.getElementById("orientation").innerHTML = "Alpha: " +
                 parseInt(event.alpha) + "</br> Beta: " + parseInt(event.beta) +
                 "</br> Gamma: " + parseInt(event.gamma);
         }, true);
      </script>
   </body>
</html>
```

Figure 8: HTML5 Code Example for Orientation

3.10 Microphone

The microphone feature is fully supported by the HTML5 standard via multiple APIs. The earlier established HTML5 API that can access the microphone is the media capture API that overrides the HTML input element to accept audio files captured straight from the microphone on mobile devices [Kostiainen 2013a]. However, this implementation is sub-optimal because during the test runs, using this API simply resulted in a file selector instead of accessing the mobile device's microphone. This means that if the mobile device does not have a voice recorder application installed, there is no way to access the microphone. The newer HTML5 API that can access the microphone is WebRTC, which can directly access the microphone and receive the voice as streaming data [Google 2011].

Figure 9 shows the code example that lets the user access the microphone and record audio to a file using the media capture API. The code example works as follows: when the user taps on the button, an application selector pops up asking the user to select an audio recorder app to record the audio file. An example of voice capture using WebRTC can be found on the HTML5Rocks website; the website provides a demo of a video chat client that resides inside a web browser with no extra plugin installations required [Dutton 2012].

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Microphone</title>
   </head>
   <body>
      <p>
         Currently it doesn't work so well with Nexus 7 Chrome browser. It just launches a
         file selector, and audio recorder is not one of the options if you don't have it
         installed.
      </p>
      <form action="server.cgi" method="post" enctype="multipart/form-data">
         <input type="file" accept="audio/*;capture=microphone">
      </form>
      <form action="server.cgi" method="post" enctype="multipart/form-data">
         <input type="file" name="audio" accept="audio/*" capture>
         <input type="submit" value="Upload">
      </form>
   </body>
</html>
```

Figure 9: HTML5 Code Example for Microphone

3.11 Compass

The compass feature is also fully supported by the HTML5 standard as it uses the same API that is used for accessing the orientation of the device. To access the compass features on mobile devices, developers can employ the already available device orientation API [Block 2011], and inversing the alpha variable will provide the heading of the compass reading.

Figure 10 shows the code example that demonstrates the compass feature. It displays the compass heading (North, South, East, or West) for the top of the device when held in portrait mode.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Compass</title>
   </head>
   <body>
      <p>
      <h1 id="compass"></h1>
      </br>
      Note: the device needs to be in portrait mode, and preferably lying on a flat surface.
      </p>
      <script>
         window.addEventListener("compassneedscalibration", function(event) {
               alert('Your compass needs calibrating! Wave your device in a figure-eight
                        motion');
               event.preventDefault();
         }, true);

         window.addEventListener("deviceorientation", function(event) {
               // process event.alpha, event.beta and event.gamma
               document.getElementById("compass").innerHTML = "You are facing: " +
                  headingToDirection(parseInt(event.alpha));
         }, true);

         //simply take 360 degrees and subtract the device orientation's alpha to get the
         // direction
         function headingToDirection(alpha) {
               var heading = 360 - parseInt(alpha);

               //approximate the heading
               if (heading > 337 || heading < 23)
                     return "North";
               if (heading > 22 && heading < 68)
                     return "North-East";
               if (heading > 67 && heading < 113)
                     return "East";
               if (heading > 112 && heading < 158)
                     return "South-East";
               if (heading > 157 && heading < 203)
                     return "South";
               if (heading > 202 && heading < 248)
                     return "South-West";
               if (heading > 247 && heading < 293)
                     return "West";
               if (heading > 292 && heading < 338)
                     return "North-West";
         }
      </script>
   </body>
</html>
```

Figure 10: HTML5 Code Example for Compass

3.12 Ambient Light

The ambient light feature is not currently supported by the HTML5 standard because, as of December 2013, W3C has only begun to draft the Ambient Light Events API [Turner 2013], and only some of the mobile web browsers have adopted it. For example, Firefox for Android supports the ambient light feature but Google Chrome and Dolphin do not.

Figure 11 shows the code example that demonstrates the ambient light feature. It reads the ambient light sensor's data (in unit of lux on the Nexus 7) and displays it in the "Ambient Light" HTML element. During the test runs, Firefox for Android was able to access and display the ambient light sensor's data, but nothing was accessed or displayed on the Google Chrome and Dolphin web browsers.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Ambient Light</title>
   </head>
   <body>
      <p>
      <h1 id="Ambient Light">Your browser does not support this feature.</h1>
      </p>
      <script>
         deviceLightHandler = function(event) {
               document.getElementById("Ambient Light").innerHTML = "Ambient light value = " +
                  event.value + " lux.";
         }

         window.addEventListener('devicelight', deviceLightHandler);

      </script>
   </body>
</html>
```

Figure 11: HTML5 Code Example for Ambient Light

3.13 Proximity

The proximity feature is not currently supported by the HTML5 standard because W3C has only began to draft the Proximity Events API as of October 2013 [Kostiainen 2013b], and only some of the mobile web browsers have adopted it as of December 2013. For example, Firefox for Android supports the proximity feature, but Google Chrome and Dolphin do not.

Figure 12 shows the code example that demonstrates the proximity feature. It shows a non-zero value when a physical object is far from the mobile device and a zero value when a physical object is close to the mobile device.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Proximity</title>
   </head>
   <body>
      <p>
      <h1 id="Proximity">Your browser does not support this feature.</h1>
      </p>
      <script>
         // Event Handler
         deviceProximityHandler = function(event) {
               document.getElementById("Proximity").innerHTML = "min = " + event.min +
                 "; max = " + event.max + "value = " + event.value;
         }
         // Assigning the Event Handler to a Listener
         window.addEventListener('deviceproximity', deviceProximityHandler);
      </script>
   </body>
</html>
```

Figure 12: HTML5 Code Example for Proximity

3.14 Ambient Temperature

Ambient temperature is not currently supported by the HTML5 standard because the draft of the Ambient Temperature Events API is only in the early stages as of October 2013 [Caceres 2013a]. The test mobile web browsers did not support it as of December 2013.

Figure 13 shows the code example that demonstrates the ambient temperature feature. It is supposed to show the ambient temperature of the device's surrounding environment in units of Celsius. However, nothing is accessed or displayed on the HTML page when executing the code example on all the test mobile web browsers.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Temperature</title>
   </head>
   <body>
      <p>
      <h1 id="Temperature">Your browser does not support this feature.</h1>
      </p>
      <script>
         tempHandler = function(event) {
               document.getElementById("Temperature").innerHTML = "Temperature = " +
                   event.value + " C";
         }

         window.addEventListener('ambienttemperature', tempHandler);
         navigator.system.watch("AmbientTemperature", tempHandler);
      </script>
   </body>
</html>
```

Figure 13: HTML5 Code Example for Ambient Temperature

3.15 Ambient Humidity

Ambient humidity is not currently supported by the HTML5 standard because the draft of the Ambient Humidity Events API is only in the early stages as of December 2013 [Caceres 2013b]. The test mobile web browsers did not support it as of December 2013.

Figure 14 shows the code example that demonstrates the ambient humidity feature. It is supposed to show the ambient humidity of the device's surrounding environment in percentage of relative humidity. However, nothing is accessed or displayed on the HTML page when executing the code example on all the test mobile web browsers.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Humidity</title>
   </head>
   <body>
      <p>
      <h1 id="Humidity">Your browser does not support this feature.</h1>
      </p>
      <script>
         humidityHandler = function(event) {
               document.getElementById("Humidity").innerHTML = "Humidity = " + event.value +
                 " %";
         }

         window.addEventListener('ambienthumidity', humidityHandler);
      </script>
   </body>
</html>
```

Figure 14: HTML5 Code Example for Ambient Humidity

3.16 Atmospheric Pressure

Atmospheric pressure is not currently supported by the HTML5 standard because the draft of the Atmospheric Pressure Events API is only in the early stages as of December 2013 [Tran 2013]. The test mobile web browsers did not support it as of December 2013.

Figure 15 shows the code example that demonstrates the ambient pressure feature. It is supposed to show the atmospheric pressure of the device's surrounding environment. However, nothing is accessed or displayed on the HTML page when executing the code example on all the test mobile web browsers.

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Pressure</title>
   </head>
   <body>
      <p>
      <h1 id="Pressure">Your browser does not support this feature.</h1>
      </p>
      <script>
         pressureHandler = function(event) {
               document.getElementById("Pressure").innerHTML = "Pressure = " + event.value + "
kP";
         }

         window.addEventListener('AtmPressure', pressureHandler);
         navigator.system.watch("AmbientAtmosphericPressure", pressureHandler);
      </script>
   </body>
</html>
```

Figure 15: HTML5 Code Example for Atmospheric Pressure

3.17 Battery Status

Battery status is not currently supported by the test mobile web browsers on the Android platform as of December 2013, even though W3C has a candidate recommendation for the battery status API as of the year 2012 [Kostiainen 2012].

The code example shown in Figure 16 is supposed to show the battery charge state, battery level, and discharge time, but all values show up as unknown when executed in the Google Chrome and Dolphin web browsers. On Firefox for Android, the battery level and discharging time do appear, but only where there is change in the battery level; however, the charging state still appears as unknown.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Battery Status API Example</title>
      <script>
         var battery = navigator.battery;

         battery.onchargingchange = function() {
               document.querySelector('#charging').textContent = battery.charging ?
                  'charging' : 'not charging';
         };

         battery.onlevelchange = function() {
               document.querySelector('#level').textContent = battery.level;
         };

         battery.ondischargingtimechange = function() {
               document.querySelector('#dischargingTime').textContent =
                  battery.dischargingTime / 60;
         };

      </script>
   </head>
   <body>
      <div id="charging">
         (charging state unknown)
      </div>
      <div id="level">
         (battery level unknown)
      </div>
      <div id="dischargingTime">
         (discharging time unknown)
      </div>
   </body>
</html>
```

Figure 16: HTML5 Code Example for Battery Status

3.18 Audio Playback

Audio playback is fully supported by the HTML5 standard. Developers can simply use the "audio" and "source" HTML elements to indicate source location of the audio file and the type of audio file to play. The mobile web browser will render an audio player when the web page opens.

Figure 17 shows the code example that demonstrates the audio playback feature. The test web browsers will render the audio element with an audio player that can control the playback of the audio file.

```
<!DOCTYPE html>
<html>
   <body>
      <audio controls>
         <source src="justin_bieber_beauty_and_a_beat.mp3" type="audio/mpeg">
         Your browser does not support the audio element.
      </audio>
   </body>
</html>
```

Figure 17: HTML5 Code Example for Audio Playback

3.19 Video Playback

Video playback is fully supported by the HTML5 standard. Developers can simply use the "video" and "source" HTML elements to indicate the source location of the video file and the type of video file to play. The mobile web browser will render a video player when the web page opens.

Figure 18 shows the code example that demonstrates the video playback feature. The test mobile web browsers will render the video element with a video player that can control the playback of the video file.

```
<!DOCTYPE html>
<html>
   <body>
      <video width="320" height="240" controls>
         <source src="Dubstep_Cat.mp4" type="video/mp4">
         Your browser does not support the video tag.
      </video>
   </body>
</html>
```

Figure 18: HTML5 Code Example for Video Playback

3.20 Camera

Accessing the camera is fully supported by the HTML5 standard via the media capture API [Kostiainen 2013a] or WebRTC [Google 2011]. Using the media capture API, developers can use the input HTML element and indicate the type of media to capture to access the camera. Using WebRTC, developers can use the application to directly access the camera as streaming data.

Figure 19 shows the code example that demonstrates the camera feature. On the selected test mobile web browsers, it displays a file selector that will prompt the user to select an app that can view images, and one of the options will be the camera app. After the user has taken a picture, the web page will display the result.

However, the problem with the media capture API is that if the mobile device does not have a camera app installed, then using the media capture API will not allow the user to access the camera.

An example of camera capture using WebRTC can be found on the HTML5Rocks website where there is a demo of a video chat client that resides inside a web browser with no extra plugin installations required [Dutton 2012].

```
<!DOCTYPE HTML>
<html>
   <head>
      <title>Camera</title>
   </head>
   <body>
      <p>
         Take a picture:
         <input type="file" name="Camera" accept="image/*" capture>
      </p>
      <canvas></canvas>
      <script>
         var input = document.querySelector('input[type=file]');

         input.onchange = function() {
               var file = input.files[0];

               displayAsImage(file);
         };

         function displayAsImage(file) {
               var imgURL = URL.createObjectURL(file), img = document.createElement('img');

               img.onload = function() {
                       URL.revokeObjectURL(imgURL);
               };

               img.src = imgURL;
               document.body.appendChild(img);
         }
      </script>
   </body>
</html>
```

Figure 19: HTML5 Code Example for Using the Camera

3.21 Vibration

Vibration is not currently supported by the HTML5 standard, even though W3C has a candidate recommendation for the Vibration API as of July 2013 [Kostiainen 2013c]. During tests, the Google Chrome and Dolphin web browsers did not support this feature, but the Firefox for Android browser did support this feature.

Figure 20 shows the code example that demonstrates the vibration feature. It shows a single button on a web page that will vibrate the mobile device for one second when the user taps the button.

```
<!DOCTYPE html>
<html>
   <body>
      <button type="button" onclick="vibrate()">
      Vibrate
      </button>
   </body>
   <script>
      function vibrate() {
        // vibrate for 1000 ms
        navigator.vibrate(1000);
      }
   </script>
</html>
```

Figure 20: HTML5 Code Example for Vibration

4 Follow-Up Evaluation: Bridging Frameworks

In Section 4, we discussed HTML5 support for edge application development features and noted that many of them were not currently supported by the HTML5 standard. As a workaround for those development features, we have investigated the use of bridging frameworks to create hybrid mobile applications. More specifically, we investigated the usage of PhoneGap to create a notification application and a battery status application for the Android platform. These two features were selected because they are not supported by the HTML5 standard, but they can be supported by using PhoneGap. We also looked at SenchaTouch as an alternative bridging framework.

4.1 PhoneGap

PhoneGap is an open-source mobile application bridging framework, based on project Cordova, that enables developers to use standard web technologies such as HTML5, CSS3, and JavaScript, to develop hybrid mobile applications across multiple mobile platforms such as iOS and Android [PhoneGap 2014a]. The hybrid mobile applications that are generated by PhoneGap appear as native applications to the mobile platforms because PhoneGap creates a single Android Activity application (i.e., wrapper application) that uses the Activity's onCreate() method to load an HTML5 web page containing all the application logic and presentation. Figure 21 shows the wrapper application source code for the notification demo application for the Android platform.

```
//NotificationDemo.java
package org.apache.cordova.notification;

import android.os.Bundle;
import org.apache.cordova.*;

public class NotificationDemo extends DroidGap
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        // Set by <content src="index.html" /> in config.xml
        super.loadUrl(Config.getStartUrl());
        //super.loadUrl("file:///android_asset/www/index.html")
    }
}
```

Figure 21: PhoneGap Wrapper Application for the Notification Demo

The main benefit of this approach is that the hybrid applications can access the mobile platform's native APIs via the PhoneGap framework. However, developers must be aware that the hybrid mobile applications are applications that appear native to the mobile platforms and must be installed onto the device like any other native mobile application. For example, the hybrid mobile applications created via PhoneGap are compiled into .apk files for the Android platform.

The following sections explain the structure and details of the hybrid mobile applications created using PhoneGap.

4.1.1 Notification Hybrid Mobile Application

Notification is an essential feature on mobile devices for notifying the user of important events. Examples of notification methods are pop-up alerts, notification bar messages, sound alerts, and vibration alerts. Unfortunately, as noted earlier, the HTML5 technology does not support this development feature. PhoneGap has a Notifications API that enables the hybrid mobile application to directly access the notification mechanisms that are available on the mobile platforms [PhoneGap 2014c].

As shown in Figure 21, the wrapper application for the notification hybrid mobile application loads the core application logic and presentation from the HTML5 web page, file:///android_asset/www/index.html that is shown in Figure 22. This HTML5 web page starts the application with the PhoneGap framework and connects it to all the PhoneGap APIs. Next, three buttons are created on the web page for showing a pop-up notification alert, playing a notification beep sound, and vibrating the mobile device. The API calls that perform these actions—navigator.notification.alert(), navigator.notification.beep(), and navigator.notification.vibrate()—are specifically defined by the PhoneGap API and not the HTML5 standard. The full set of PhoneGap APIs is available on the PhoneGap Documentation website [PhoneGap 2014b].

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <meta name="format-detection" content="telephone=no" />
      <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1,
           minimum-scale=1, width=device-width, height=device-height,
           target-densitydpi=device-dpi" />
      <link rel="stylesheet" type="text/css" href="css/index.css" />
      <title>Notification Demo</title>
   </head>
   <body>
      <div class="app">
         <h1>Apache Cordova</h1>
         <div id="deviceready" class="blink">
            <p class="event listening">
               Connecting to Device
            </p>
            <p class="event received">
               Device is Ready
            </p>
            <button type="button" onclick="showAlert(); return false;">
            <font size="7">Show Alert</font>
            </button>
            <br/>
            <br/>
            <br/>
            <button type="button" onclick="playBeep(); return false;">
            <font size="7">Play Beep</font>
            </button>
            <br/>
            <br/>
            <br/>
            <button type="button" onclick="vibrate(); return false;">
            <font size="7">Vibrate</font>
            </button>
         </div>
      </div>
      <script type="text/javascript" src="cordova.js"></script>
      <script type="text/javascript" src="js/index.js"></script>
      <script type="text/javascript">
         app.initialize();

         // alert dialog dismissed
         function alertDismissed() {
         // do nothing for now }

         // Show a custom alert
         function showAlert() {
               navigator.notification.alert(
               'You are the champion!', // message
               alertDismissed,          // callback
               'Success',               // title
               'Rejoice'                // buttonName);
         }

         // Beep 1 time
         function playBeep() {
               navigator.notification.beep(1);}

         // Vibrate for 2 seconds
         function vibrate() {
               navigator.notification.vibrate(2000); }
      </script>
   </body>
</html>
```

Figure 22: Notification Application Logic and Presentation HTML5 Web Page

4.1.2 Battery Status Hybrid Mobile Application

Battery status is an important development feature that can enable applications to tailor the application's level of processing and resource usage based on the remaining battery life of a user's session. HTML5 alone cannot access this information, but there is a PhoneGap API that provides battery status information [PhoneGap 2014d].

Figure 23 shows the core application logic and presentation code for the battery status hybrid mobile application. It attaches the battery listener upon app initialization, and when it receives the battery information event, it posts a message indicating remaining battery percentage and whether the device is plugged into a power source.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <meta name="format-detection" content="telephone=no" />
      <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1,
          minimum-scale=1, width=device-width, height=device-height,
          target-densitydpi=device-dpi" />
      <link rel="stylesheet" type="text/css" href="css/index.css" />
      <title>Hello World</title>
   </head>
   <body>
      <div class="app">
         <h1>Apache Cordova</h1>
         <div id="deviceready" class="blink">
            <p class="event listening">
               Connecting to Device
            </p>
            <p class="event received">
               Device is Ready
            </p>
         </div>
      </div>
      <br />
      <br />
      <br />
      <h1 id='BatteryInfo'>Battery information has not been retrieved yet.</h1>
      <script type="text/javascript" src="cordova.js"></script>
      <script type="text/javascript">
         var app = {
               // Application Constructor
               initialize : function() {
                      this.bindEvents();
               },
               // Bind Event Listeners
               // Bind any events that are required on startup. Common events are:
               // 'load', 'deviceready', 'offline', and 'online'.
               bindEvents : function() {
                      document.addEventListener('deviceready', this.onDeviceReady, false);
               },
               // deviceready Event Handler
               // The scope of 'this' is the event. In order to call the 'receivedEvent'
               // function, we must explicitly call 'app.receivedEvent(...);'
               onDeviceReady : function() {
                      app.receivedEvent('deviceready');
                      //attach the battery listener here
                      window.addEventListener("batterystatus", onBatteryStatus, false);
               },
               // Update DOM on a Received Event
               receivedEvent : function(id) {
                      var parentElement = document.getElementById(id);
                      var listeningElement = parentElement.querySelector('.listening');
                      var receivedElement = parentElement.querySelector('.received');
                      listeningElement.setAttribute('style', 'display:none;');
                      receivedElement.setAttribute('style', 'display:block;');
                      console.log('Received Event: ' + id);
               }
         };
         function onBatteryStatus(info) {
               // Handle the online event
               document.getElementById('BatteryInfo').innerHTML = "Is battery plugged? " +
                  info.isPlugged + ". The battery Level is: " + info.level + "%";
         }
         app.initialize();
      </script>
   </body>
</html>
```

Figure 23: Battery Status Application Logic and Presentation HTML5 Web Page

4.2 SenchaTouch

SenchaTouch is another bridging framework that uses web technologies to develop cross-platform hybrid mobile applications. The main focus of SenchaTouch is the user interface (UI) of hybrid mobile applications, which is why it provides an extensive set of UI elements for developers. Applications implemented using SenchaTouch still have the portability aspect because they can run on multiple mobile platforms [Sencha 2013a].

The SenchaTouch SDK includes many examples that demonstrate capabilities, and they all use current HTML5 web technologies. The basic development process is to implement the hybrid mobile applications using SenchaTouch APIs and then upload them to a web server. The user can access those apps by visiting the website/server [Pearce 2011].

Though applications written using SenchaTouch can exist as mobile web applications, they cannot access the device-level APIs more than the HTML5 technology alone can. In order to access the device-level APIs, such as the notification API, the developer still must integrate PhoneGap with SenchaTouch and package the applications into native mobile applications using PhoneGap [Sencha 2013b].

5 Software Architecture Implications

Aside from the functional requirements for edge applications, there are software quality attributes to consider, such as the maintainability, performance, and portability of the applications. The main focus of this section is to examine the effect that the use of HTML5 and bridging frameworks has on these quality attributes.

5.1 Maintainability

Maintainability is defined as "the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers" [Bass 2013]. We are specifically interested in the ease of updating the applications as technologies evolve. The questions to answer are the following:

- Will the evolution of HTML5 affect currently implemented HTML5 mobile applications?
- Will the evolution of Android OS[4] affect currently implemented HTML5 mobile applications?
- Will the evolution of the bridging frameworks affect currently implemented hybrid mobile applications?
- Can hybrid mobile applications be migrated to HTML5 mobile applications as new development features are added to the HTML5 standard?

If the answer to any of the above question is "yes," then the follow up question is "What percentage of code must be updated and how much development time will it consume?"

The following subsections will explore the answers to these questions in detail.

5.1.1 Evolution of HTML5 and Its Effects on HTML5 Applications

The evolution of HTML5 involves adding new APIs to the specification and enabling more development features. As listed in Table 2 of Section 4, there are many development features that are not currently supported by the HTML5 standard but are in the process of being formalized.

In looking at the changes to the HTML5 specification over the years, we notice that once an API has been formally approved, there are no further changes to the API. For example, the Geolocation API has not changed from the year 2008 draft [Propescu 2008] to the current approved specification dated 2013 [Propescu 2012]. The same navigator.geolocation.watch Position() JavaScript function is used to track the location of the mobile device. If this practice continues, then the evolution of HTML5 will not affect any existing mobile applications implemented using the HTML5 standard.

5.1.2 Evolution of Android OS and Its Effects on HTML5 Applications

To understand the effects of Android OS evolution on HTML5 applications, we looked at the version histories of Android OS. Table 3 shows the only two entries that relate to HTML5 support and their corresponding release dates. HTML5 support was first provided in Android 2.0 Éclair,

[4] The question is limited to Android OS because that was the platform that was selected for the experiments. However, the same question would apply to any other mobile platform.

which was released on October 26, 2009. The next entry is from Android 2.2 Froyo, which added the HTML5 Files API support but did not change any of the existing HTML5 support.

Table 3: Android OS Version History Related to HTML5 Support [Wikipedia 2013a]

Date	Version Number	Change Log Details
26 Oct. 2009	Android 2.0 Éclair	refreshed browser UI with bookmark thumbnails, double-tap zoom and support for HTML5 (Initial support for HTML5 on Android)
20 May 2010	Android 2.2 Froyo	support for file upload fields in the Browser application (Files API support)

Because HTML5 support is more likely to be built into the web browsers, we examined the version histories of two mobile web browsers on the Android platform: Google Chrome and Firefox for Android. Table 4 shows all the versions of Google Chrome that relate to HTML5 support and their corresponding release dates. As with the Android OS version history, we noticed that in all the versions of Google Chrome, only new HTML5 features were added and no changes were made towards existing HTML5 support.

Table 4: Android Google Chrome Version History Related to HTML5 Support [Wikipedia 2013b]

Date	Version Number	Change Log Details
12 Oct. 2009	3.0.195	HTML5 video and audio tag support
25 Jan. 2010	4.0.249	improved HTML5 support
21 May 2010	5.0.375	increased HTML5 support (Geolocation API, App Cache, web sockets, and file drag-and-drop)
21 Oct. 2010	7.9.517	implemented HTML5 parsing algorithm, File API
27 Apr. 2011	11.0.696	HTML5 Speech Input API
31 Jul. 2012	21.0.1180	HTML5 audio/video and WebAudio now support 24-bit PCM wave files
10 Jan. 2013	24.0.1312	The HTML5 datalist element now supports suggesting a date and time.
2 Oct. 2013	30.0.1599	DeviceMotion (device acceleration and rotation rates) events and two new experimental features behind a flag: Web Speech API (recognition) and the Vibration API

Table 5 shows all the versions of Firefox for Android that relate to HTML5 support and their corresponding release dates. Similar to the Android OS version history as well as the Google Chrome version history, we noticed that in all the versions of Firefox for Android, only new HTML5 features were added and no changes were made towards existing HTML5 support.

Table 5: Firefox for Android Version History Related to HTML5 Support [Wikipedia 2013c]

Date	Version Number	Change Log Details
29 Mar. 2011	4	HTML5 support in Firefox for Android and Maemo includes Location-Aware Browsing, device orientation, accelerometer, desktop notifications and more
16 Aug. 2011	6	single touch events API, and IndexedDB API for local database storage
27 Sep. 2011	7	The WebSocket API is now available in Firefox for Android.
21 Dec. 2011	9	HTML5 Input Tag for Camera Access, and HTML5 Form Validation
31 Jan. 2012	10	HTML5 new <bdi> element for bi-directional text isolation, along with supporting CSS properties
16 Feb. 2012	11	HTML5 The outerHTML property is now supported on HTML elements.
28 Aug. 2012	15	HTML5 The <audio> and <video> elements now support the played attribute, and the <source> element now supports the media attribute.
2 Apr. 2013	20	HTML5 <canvas> now supports blend modes, and Various <audio> and <video> improvements.
25 Jun. 2013	22	new HTML5 <data> and <time> elements

From the examination of the version histories of Android OS, Google Chrome web browser, and Firefox for Android web browser, and observing that no changes were implemented on any existing HTML5 feature support, we concluded that if these practices continue, the evolution of Android OS will not affect any existing mobile application implemented using the HTML5 standard. However, this conclusion is simply based on historical data and does not preclude future changes in practices.

5.1.3 Evolution of Bridging Frameworks and Its Effects on Hybrid Mobile Applications

To understand the evolution of bridging frameworks and its effect on hybrid mobile applications, we examined the version history of PhoneGap, looking for any major differences in its APIs. PhoneGap was selected simply because of its larger web presence, its open source, and its easily available documentation.

From the oldest version of PhoneGap (version 0.9.2)[PhoneGap 2014e] up to the newest version of PhoneGap (version 3.1.0) [PhoneGap 2014b], new APIs were added in each subsequent release, and there was only one API that was changed during this time frame: The Network API from version 0.9.5.1 [PhoneGap 2014f] was changed to the Connection API in version 0.9.6 [PhoneGap 2014g]. This API is mainly used to retrieve the information of the connectivity meth-

ods available on the mobile device—for example, to determine whether the device has cellular or Wi-Fi connectivity [PhoneGap 2014h].

From examining the version history of PhoneGap, we concluded that the evolution of bridging frameworks will not affect any existing hybrid mobile applications developed using PhoneGap, unless the applications used the Network API from version 0.9.5.1 or lower and wanted to switch to using PhoneGap version 0.9.6 or higher.

5.1.4 Migrating Hybrid Mobile Applications to HTML5 Applications

To understand the difficulty of transitioning a hybrid mobile application to an HTML5 application, we selected the accelerometer feature because it could be developed using either PhoneGap or HTML5.

Figure 24 shows the core application logic and presentation code for the accelerometer hybrid mobile application. There is a single button on the HTML page, and when it is pressed, it displays the accelerometer data exactly as in the accelerometer HTML5 mobile application shown in Section 4.8, Figure 7. This code example will work if it is built and deployed with PhoneGap, but will not run properly and will show errors if it is opened in any web browser. This result is expected because the PhoneGap APIs and HTML5 APIs are very different. In addition, the web browser is not able to detect the PhoneGap APIs without building the code through PhoneGap because the app cannot connect to the PhoneGap framework during the app initialization stage (the first line of the embedded JavaScript: app.initialize()).

```
<html>
   <head>
      <meta charset="utf-8" />
      <meta name="format-detection" content="telephone=no" />
      <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1,
           minimum-scale=1, width=device-width, height=device-height,
           target-densitydpi=device-dpi" />
      <link rel="stylesheet" type="text/css" href="css/index.css" />
      <title>Acceleration</title>
   </head>
   <body>
      <div class="app">
         <h1>Apache Cordova</h1>
         <div id="deviceready" class="blink">
            <p class="event listening">
               Connecting to Device
            </p>
            <p class="event received">
               Device is Ready
            </p>
         </div>
      </div>
      <br/><br/><br/><br/>
      <button type="button" onclick="toggleAccel();">
      <font size="10">Toggle Accelerometer</font>
      </button>
      <br/><br/><br/><br/>
      <h1>Current acceleration in x-coordinate: </h1>
      <h1 id='x'></h1>
      <br/><br/><br/><br/>
      <h1>Current acceleration in y-coordinate: </h1>
      <h1 id='y'></h1>
      <br/><br/><br/><br/>
      <h1>Current acceleration in z-coordinate: </h1>
      <h1 id='z'></h1>
      <script type="text/javascript" src="cordova.js"></script>
      <script type="text/javascript" src="js/index.js"></script>
      <script type="text/javascript" src="main.js"></script>
      <script type="text/javascript">
         app.initialize();
         var accelerationWatch = null;
         function updateAcceleration(a) {
               document.getElementById('x').innerHTML = roundNumber(a.x);
               document.getElementById('y').innerHTML = roundNumber(a.y);
               document.getElementById('z').innerHTML = roundNumber(a.z);
         }
         var toggleAccel = function() {
               if (accelerationWatch !== null) {
                     navigator.accelerometer.clearWatch(accelerationWatch);
                     updateAcceleration({
                           x : "", y : "", z : ""
                     });
                     accelerationWatch = null;
               } else {
                     var options = {};
                     options.frequency = 1000;
                     accelerationWatch =
                           navigator.accelerometer.watchAcceleration(updateAcceleration,
                           function(ex) {
                             alert("accel fail (" + ex.name + ": " + ex.message + ")");
                     }, options);
               }
         };
      </script>
   </body>
</html>
```

Figure 24: Accelerometer Application Logic and Presentation HTML5 Web Page

However, the migration from a hybrid mobile application to a pure HTML5 mobile application is still possible. The developer would have to manually remove all the JavaScript code related to PhoneGap, and replace it with the corresponding HTML5 JavaScript code. Although up to 100% of the JavaScript source code of a hybrid mobile app created using PhoneGap must be replaced, the original HTML web page layout and the CSS files that define the look and feel of the web page can be preserved.

In summary, to migrate hybrid mobile applications to HTML5 mobile applications, the application logic must be modified, but the presentation of the mobile application can be preserved. Having a coding standard that localizes JavaScript code in specific areas of the HTML5 file or places it in external files can facilitate the process.

5.2 Performance

Performance is about "time and a software system's ability to meet timing requirements" [Bass 2013]. We are specifically interested in two aspects related to performance: application execution time and memory usage. Though there are many articles that indicate that native applications have better performance than HTML5 applications [Regmi 2011], we still conducted a simple performance test to validate this statement: We measured the application execution time (one run) and memory usage of a native mobile application and an HTML5 mobile application that use the Geolocation feature.

5.2.1 Execution Time Analysis

To compare the execution time of a native mobile application against an HTML5 mobile application using the Geolocation feature, timestamp code was added at the beginning of the mobile application and then subtracted from the timestamp after the location of the mobile device was retrieved. The measured execution time does not include the HTML page fetching and rendering, as it only measures the time between when the Geolocation API is called and when a location is received. In addition, because the execution time may fluctuate due to the mobile device background processes or other operating system factors, three test runs were performed for each mobile application and their average time was retrieved for comparison. Also, network location is used over GPS location for more consistent timings. Our test runs were performed on the Samsung Galaxy S4, and Table 6 lists the results of the test runs for a native mobile application and an HTML5 mobile application that solely use the Geolocation feature upon application startup.

Table 6: Execution Times for a Native and an HTML5 Mobile Geolocation Application

Test Run	Native Mobile Application Execution Time (milliseconds)	HTML5 Mobile Application Execution Time (milliseconds)
#1	59	192
#2	64	282
#3	67	167
Average	63	214

Our test runs show that accessing the Geolocation feature on a native mobile application is approximately three times faster than accessing the Geolocation feature on an HTML5 native mobile

application, which supports the claim that HTML5 mobile applications are slower in terms of execution time when compared to native mobile applications. This is expected, given that HTML5 applications run within a web browser.

5.2.2 Memory Usage Analysis

To understand the memory usage of native mobile applications and HTML5 applications, we used a free memory usage analysis tool on the Android mobile device called Memory Usage[5] to display the memory usage of any running process. Measuring the memory usage of the native mobile application is easy: we observe the memory usage of the running application. However, measuring the memory usage of an HTML5 application is more complex because the HTML5 mobile application runs inside a web browser. In our tests we first measured the memory usage of the mobile web browser (Google Chrome and Firefox for Android) with no web pages open to obtain a measurement baseline. We then measured the memory usage of the web browser running only the HTML5 mobile application that uses the Geolocation feature. Finally we subtracted the first value (only web browser) from the second value (browser plus mobile application) to obtain an approximate measurement of the memory usage for just the HTML mobile application. Table 7 shows all these values.

Table 7: Memory Usage for a Native and an HTML5 Mobile Geolocation Application

Software Component	Memory Usage (MB)
(1) Native Mobile Application	4.4
(2) Google Chrome (no web pages open)	11.16
(3) HTML5 Mobile Application Running in Google Chrome (Total)[6]	24.16
(4) HTML5 Mobile Application Running in Google Chrome (Only app: (3) – (2))	13.0
(5) Firefox for Android (no web pages open)	11.1
(6) HTML5 Mobile Application Running in Firefox for Android (Total)	25.1
(7) HTML5 Mobile Application Running in Firefox for Android (Only app: (6) – (5))	14.0

[5] https://play.google.com/store/apps/details?id=mem.usage

[6] The memory usage values for Google Chrome are the result of adding the memory usage values of the two running processes associated to Google Chrome

Our analysis shows that the memory usage of a native mobile application is approximately three times smaller than an HTML5 mobile application that uses the Geolocation feature, which supports the claim that HTML5 mobile applications are less memory efficient when compared with native mobile applications. It is interesting to note that the memory usage of the HTML5 mobile application does not vary greatly between the two mobile web browsers. However, given that we used a very simple method to determine the amount of memory consumed by the HTML5 mobile application, the results are likely not precise.

5.3 Portability of HTML5 Among Mobile Web Browsers

Portability refers to "the ease with which software that was built to run on one platform can be changed to run on a different platform" [Bass 2013**]**. Portability is one of the drivers of HTML5 that enable applications to be developed once and then to run on multiple mobile platforms. However, in our studies, we were interested in examining the portability of HTML5 applications across the selected test mobile web browsers.

During the investigation and implementation of HTML5 mobile applications, we noticed that different web browsers support HTML5 development features differently. Table 8 lists the edge application development features and their support from the selected test mobile web browsers. The HTML5 feature support is listed either as "fully supported" if the HTML5 code example for that particular feature works flawlessly or "not supported" if the HTML5 code example for that particular feature did not work as of December 2013.

Table 8: HTML5 Features Support for Android Web Browsers

Edge Application Development Feature	Google Chrome for Android	Firefox for Android	Dolphin for Android
geolocation	fully supported	fully supported	fully supported
Wi-Fi communication – client-server	fully supported	fully supported	not supported
Wi-Fi communication – peer-to-peer	fully supported	fully supported	not supported
Bluetooth communication	not supported	not supported	not supported
background processing	fully supported	fully supported	not supported
interactive UI (such as touch interface and gestures)	fully supported	fully supported	not supported
user notification	not supported	not supported	not supported
access to files	fully supported	fully supported	fully supported
accelerometer	fully supported	fully supported	not supported
orientation	fully supported	fully supported	fully supported
microphone	fully supported	fully supported	not supported
compass	fully supported	fully supported	fully supported
ambient light	not supported	fully supported	not supported

proximity	not supported	fully supported	not supported
ambient temperature	not supported	not supported	not supported
ambient humidity	not supported	not supported	not supported
atmospheric pressure	not supported	not supported	not supported
battery status	not supported	fully supported	not supported
audio playback	fully supported	fully supported	fully supported
video playback	fully supported	fully supported	fully supported
camera	fully supported	fully supported	fully supported
vibration	not supported	fully supported	not supported

From testing the HTML5 features on multiple test mobile web browsers, we are able to conclude that the support of HTML5 features across mobile web browsers is not consistent; this means that the choice of web browser is very important when running edge applications.

6 Related Work

Researchers have conducted many studies on the usefulness of HTML5 technology with in-depth analysis of many of its features. In one such study, Melamed and Clayton evaluated HTML5 features such as offline storage, geolocation, interactive user interface, and local persistence of data for designing and implementing pervasive media applications [Melamed 2010]. They found that HTML5 combined with geolocation could provide most of the features required by pervasive media mobile apps. However, they did not analyze all the features that can be leveraged by edge applications. Torunski evaluated the use of HTML5 in a cloud-based system targeted at emergency response planning in which HTML5 was used to build client software that retrieved and used information from cloud servers [Torunski 2012]. The study showed that HTML5 simplified the development of cloud-based systems. However, the study did not target the usage of HTML5 for mobile applications, but rather focused on desktop applications and rich graphical user interface features.

Many research studies have focused on using HTML5 on mobile devices to solve a specific problem. Zhu and colleagues focused on using HTML5 to build a framework for mobile web application development that improves reusability of source code and eases the reconstruction and integration of mobile applications [Zhu 2013]. Regmi and colleagues analyzed the performance of web applications implemented using HTML5, with a specific focus on network dependency, user experience, and performance of mobile devices [Regmi 2011]. Jeremić and colleagues list a broad set of HTML5 features for mobile devices and describe, step-by-step, how to use these features to build a mobile web application [Jeremic 2013]. Hu and colleagues specifically researched location-based services, using the HTML5 Geolocation feature and Google Maps application programming interfaces (APIs) [Hu 2013]. In addition to research papers, there are also many books that discuss and teach mobile HTML5 application development [Kyrnin 2012, Kessin 2012, Holdnener 2011, Oehlman 2011]. Although these studies focus on the use of HTML5 features in mobile application, they do not completely address the features required by sensor-rich and dynamic edge applications.

Finally, there is research conducted on the comparison of native and hybrid mobile application development approaches. Hybrid mobile application development approaches combine native mobile development features with HTML5 and other web application development features. Zibula and Majchrzak analyzed cross-platform development using HTML5, jQuery Mobile, and PhoneGap [Zibula 2012]. Ristimäki discusses the use of the Android software development kit (SDK) for implementing a mobile application to control sensors and actuators [Ristimaki 2013]. Ghatol and Patel describe and teach hybrid mobile application development using PhoneGap, which is a mobile web framework that provides the feature set of native mobile development to mobile web application development [Ghatol 2011]. Although these research documents provide a broad overview of using bridging frameworks to develop mobile applications in a hybrid manner, combining the use of native mobile SDKs and HTML5, they do not specifically address how hybrid mobile development approaches can benefit edge applications.

7 Summary and Conclusions

We analyzed a total of 22 edge application development features via the construction of code examples for each of the development features. Thirteen of those features are supported by HTML5, eight have high possibility of being supported in the future, and one feature is not supported by HTML5.

As for the missing development features required by edge applications, we analyzed whether bridging frameworks such as PhoneGap or SenchaTouch could fulfill the requirements. Though the bridging frameworks do provide more native device-level API access, the resulting applications appear native to the mobile platforms, which means that they must be installed on the mobile device.

Finally, we examined the software architecture implications of HTML5 mobile applications and discovered that the maintainability of HTML5 mobile applications looks promising as the evolution of the HTML5 standard and Android OS has not modified any released APIs. When we compared the performance of HTML5 mobile applications to native applications on the Android platform, we discovered that indeed the HTML5 mobile applications are 3.4 times slower and use 3 times more memory. In addition, from running our HTML5 code examples, we discovered that portability of HTML5 mobile applications across mobile web browsers varies: Firefox for Android supports 16 HTML5 edge application development features, Android Google Chrome supports 13 features, and Dolphin supports 7 HTML5 edge application development features.

Overall, the HTML5 web standard as of today still lacks support for many of the critical edge development features. Bridging frameworks can fill the missing development features, but unfortunately, applications become compiled into native mobile applications. However, there has been huge progress since this research started and adoption of HTML5 is growing. There are entire operating systems currently being developed in HTML5, such as Firefox OS.[7] Although HTML5 might not be currently ready for the development of edge applications because many required features are still not fully suppported, it will likely only be a matter of time before it is possible.

[7] http://www.mozilla.org/en-US/firefox/os/

Bibliography

URLs are valid as of the publication date of this document.

[Bass 2013]
Bass, Len, Clements, Paul & Kazman, Rick. *Software Architecture in Practice – 3rd ed*. Addison-Wesley SEI Series in Software Engineering. Boston, MA, USA. 2013.

[Bidelman 2010]
Bidelman, Eric. "Reading Files in JavaScript Using the File APIs." *HTML5 Rocks*. 18 June, 2010. http://www.html5rocks.com/en/tutorials/file/dndfiles/

[Block 2011]
Block, Steve & Popescu, Andrei. "DeviceOrientation Event Specification." *World Wide Web Consortium (W3C)*. 2011. http://dev.w3.org/geo/api/spec-source-orientation.html

[Caceres 2013a]
Cáceres, Marcos & Nv, Balaji. "Ambient Temperature Events." *World Wide Web Consortium (W3C)*. 2013. https://dvcs.w3.org/hg/dap/raw-file/default/temperature/Overview.html

[Caceres 2013b]
Cáceres, Marcos & Nv, Balaji. "Ambient Humidity Events." *World Wide Web Consortium (W3C).* 2013. https://dvcs.w3.org/hg/dap/raw-file/default/humidity/Overview.html

[Chromium 2013]
The Chromium Blog. "Chrome 29 Beta: Web Audio and WebRTC in Chrome for Android." 2013. http://blog.chromium.org/2013/07/chrome-29-beta-web-audio-and-webrtc-in.html

[Dutton 2012]
Dutton, Sam. "Getting Started with WebRTC." *HTML5 Rocks*. 23 July 2012. http://www.html5rocks.com/en/tutorials/webrtc/basics/

[Ghatol 2011]
Ghatol, Rohit & Patel, Yogesh. *Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5*. Apress, 2011. http://books.google.com/books?hl=en&lr=&id=GTXQWFtx34sC&oi=fnd&pg=PA1&dq=html5+android+application+sensors+%22html5%22+-presentation+-brochure&ots=t0E5ej60CA&sig=JdXgbOZZ9A56XlEEqNT5KsKyz-4#v=onepage&q&f=false

[Google 2011]
Google Inc. *WebRTC*. 2011-2013. http://www.webrtc.org/

[Gregg 2013]
Gregg, John & van Kesteren, Anne "Web Notifications." *World Wide Web Consortium (W3C)*. 2013. http://www.w3.org/TR/2013/WD-notifications-20130912/

[Hickson 2011]
Hickson, Ian. "The WebSocket API." *World Wide Web Consortium (W3C)*, 2011. http://www.w3.org/TR/2011/WD-websockets-20110929/

[Hickson 2012]
Hickson, Ian. "Web Workers," *World Wide Web Consortium (W3C)*. 2012. http://www.w3.org/TR/workers/

[Holdnener 2011]
Holdnener III, Anthony T. *HTML5 Geolocation by O'Reilly.* O'Reilly Media, Inc., 2011. http://books.google.com/books?hl=en&lr=&id=9aIA5P6dp2cC&oi=fnd&pg=PR7&dq=html5+android+application+sensors+%22html5%22+-presentation+-brochure&ots=PgMnv8T06T&sig=uRUqggA8uClWlfwJn_G8ci8PKn8#v=onepage&q=html5%20android%20application%20sensors%20%22html5%22%20-presentation%20-brochure&f=false

[Hu 2013]
Hu, Wen-Chen, Kaabouch, Naima, Yang, Hung-Jen & Wang, Xiwei. "Location Based Services Using HTML5 Geolocation and Google Maps APIs." http://micsymposium.org/mics_2013_Proceedings/submissions/mics20130_submission_1.pdf

[Jeremic 2013]
Jeremić, Miljan, Damjanović, Zvonko, Kostadinović, Đorđe & Jeremić, Dušan. "Build a Mobile App with HTML5 and JavaScript." *Economics Management Information Technology 1*, 4, 2013. http://emit.kcbor.net/Kompletni%20casopisi/EMIT%20Vol1%20No4.pdf#page=37

[Kessin 2012]
Kessin, Zachary. *Programming HTML5 Applications,* 1st ed. O'Reilly Media, Inc., 2012. http://books.google.com/books?hl=en&lr=&id=h8_Dylj8AigC&oi=fnd&pg=PR3&dq=programming+html5+application+oreilly&ots=KPu33kzEM6&sig=wp8J2kfNUFSEWhQdGyRinlDjxSY

[Kostiainen 2012]
Kostiainen, Anssi & Lamouri, Mounir. "Battery Status API." *World Wide Web Consortium (W3C)*. 2012. http://www.w3.org/TR/battery-status/

[Kostiainen 2013a]
Kostiainen, Anssi, Oksanen, Ilkka & Hazaël-Massieux, Dominique. "HTML Media Capture." *World Wide Web Consortium (W3C)*. 2013. http://www.w3.org/TR/html-media-capture/

[Kostiainen 2013b]
Kostiainen, Anssi & Tran, Dzung D. "Proximity Events." *World Wide Web Consortium (W3C).* 2013. http://www.w3.org/TR/proximity/

[Kostiainen 2013c]
Kostiainen, Anssi. "Vibration API." *World Wide Web Consortium (W3C).* 2013. http://www.w3.org/TR/vibration/

[Kyrnin 2012]
Kyrnin, Jennifer. *Sam's Teach Yourself HTML5 Mobile Application Development.* Pearson Education, Inc., 2012.
http://books.google.com/books?id=Zz3LwyDCL5MC&printsec=frontcover&dq=android+html5+development&hl=en&sa=X&ei=i5_RUenMLsbA4APZ94HQCw&ved=0CE0Q6AEwAw#v=onepage&q&f=true

[Melamed 2010]
Melamed, Tom & Clayton, Ben. "A Comparative Evaluation of HTML5 as a Pervasive Media Platform." *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 35* (2010): 307-325.
http://link.springer.com/chapter/10.1007/978-3-642-12607-9_20#page-1

[Mozilla 2013a]
The Mozilla Blog. "Web RTC Now Available Aross Mobile and Desktop with new Firefox with Android Compatibility." September 17, 2013. https://blog.mozilla.org/blog/2013/09/17/webrtc-now-available-across-mobile-and-desktop-with-new-firefox-for-android-compatibility

[Mozilla 2013b]
Mozilla Developer Network. Java Script. https://developer.mozilla.org/en-US/docs/Web/JavaScript (2013).

[Oehlman 2011]
Oehlman, Damon & Blanc, Sebastien. *Pro Android Web Apps: Develop for Android UsingHTML5, CSS3 & JavaScript.* Apress, 2011.
http://books.google.com/books?id=n6arJlPuiA0C&pg=PT2&lpg=PA1&ots=TG69Pn6ovr&dq=html5+android+application+development+html5&lr=

[Pearce 2011]
Pearce, James "Sencha Touch Quick Start." Sencha Inc., December 15, 2011.
http://www.sencha.com/learn/sencha-touch-quick-start/

[PhoneGap 2014a]
PhoneGap Documentation. Overview.
http://docs.phonegap.com/en/3.0.0/guide_overview_index.md.html#Overview (2014).

[PhoneGap 2014b]
PhoneGap Documentation v3.1.0. Home. http://docs.phonegap.com/en/3.1.0/index.html (2014).

[PhoneGap 2014c]
PhoneGap Documentation. Notification.
http://docs.phonegap.com/en/3.1.0/cordova_notification_notification.md.html#Notification (2014).

[PhoneGap 2014d]
PhoneGap Documentation. Events.
http://docs.phonegap.com/en/3.1.0/cordova_events_events.md.html#Notification (2014).

[PhoneGap 2014e]
PhoneGap Documentation v0.9.2 Home. http://docs.phonegap.com/en/0.9.2/index.html (2014).

[PhoneGap 2014f]
PhoneGap Documentation v0.9.5.1 Home. http://docs.phonegap.com/en/0.9.5.1/index.html (2014).

PhoneGap 2014g]
PhoneGap Documentation v0.9.6 Home. http://docs.phonegap.com/en/0.9.6/index.html (2014).

[PhoneGap 2014h]
PhoneGap Documentation Connection. http://docs.phonegap.com/en/3.1.0/cordova_connection_connection.md.html#Connection (2014).

[Propescu 2008]
Popescu, Andrei. "Geolocation API Specification." *World Wide Web Consortium (W3C)*. December 2008. http://www. w3.org/2008/TR/WD-geolocation-API-20081222

[Propescu 2012]
Popescu, Andrei. "Geolocation API Specification." *World Wide Web Consortium (W3C)*. May 2012. http://dev.w3.org/geo/api/spec-source.html

[Pusher 2012]
Pusher Ltd. *Pusher*. http://pusher.com (2012).

[Ranganathan 2013]
Ranganathan, Arun & Sicking, Jonas. "File API." *World Wide Web Consortium (W3C)*. 2013. http://www.w3.org/TR/FileAPI/

[Regmi 2011]
Regni,Saroj Sharan & Adhikari, Suyog Man Singh. *Network Performance of HTML5 Web Application in Smartphone*. Master's Thesis. School of Computing, Blekinge Institute of Technology, 2011. http://www.medieteknik.bth.se/fou/cuppsats.nsf/all/cd7caf2c8592833dc1257952007126bf/$file/BTH2011Regmi.pdf

[Ristimaki 2013]
Ristimäki, Jarno. *Android Interface to a Wireless Autonomous Wide Area Sensor Network*. Master's Thesis. Helsinki Metropolia University of Applied Sciences, 2012. http://theseus17-kk.lib.helsinki.fi/bitstream/handle/10024/42503/thesis_final_v2.pdf?sequence=1

[Schepers 2013]
Schepers, Doug, Moon, Sangwhan, Brubeck, Matt & Barstow, Arthur, eds. "Touch Events." *World Wide Web Consortium (W3C)*. 2013. http://www.w3.org/TR/touch-events/

[Sencha 2013a]
Sencha Inc. "SenchaTouch Overview." 2013. http://www.sencha.com/products/touch/

[Sencha 2013b]
Sencha Inc. “SenchaTouch Features.” 2013. http://www.sencha.com/products/touch/features

[Torunski 2012]
Torunski, Eric. *Cloud-Based Collaborative Creation and Simulation of Courses of Action: Creation of a Prototype Web Application Using New HTML5 Features.* CAE Inc., University of Ottawa, 2012.
http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6291528&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6291528

[Tran 2013]
Tran, Dzung D. “Atmospheric Pressure Events.” *World Wide Web Consortium (W3C)*, 2013.
https://dvcs.w3.org/hg/dap/raw-file/default/pressure/Overview.html

[Turner 2013]
Turner, Doug & Kostiainen, Anssi. “Ambient Light Events.” *World Wide Web Consortium (W3C)*, 2013. http://www.w3.org/TR/ambient-light/

[Wikipedia 2013a]
Wikipedia. “Android Version History.” 2013.
http://en.wikipedia.org/wiki/Android_version_history

[Wikipedia 2013b]
Wikipedia. “Google Chrome for Android.” 2013.
http://en.wikipedia.org/wiki/Google_Chrome_for_Android#Release_history

[Wikipedia 2013c]
Wikipedia. “Firefox for Mobile.” 2013. http://en.wikipedia.org/wiki/Firefox_for_mobile#History

[W3C 2013]
World Wide Web Consortium. “Cascading Style Sheets Home Page.”
http://www.w3.org/Style/CSS/Overview.en.html (2013).

[W3C 2014]
World Wide Web Consortium. “HTML 5.1 Nightly: A Vocabulary and Associated APIs for HTML and XHTML.” Editor's Draft, 21 January 2014.
http://www.w3.org/html/wg/drafts/html/master/

[Zhu 2013]
Zhu, XiaoMin, Chen, Donghua, Chen, Ying, & Chen, Huisheng. *A Resource Integration Approach for HTML5 Mobile Applications*. Springer Science+Business Media, 2013.
http://link.springer.com/article/10.1007/s10799-013-0158-9

[Zibula 2012]
Zibula, Alexander & Majchrzak, Tim A. *Cross-Platform Development Using HTML5, jQuery Mobile, and PhoneGap: Realizing a Smart Meter Application*. Springer, 2012.
http://link.springer.com/content/pdf/10.1007%2F978-3-642-36608-6_2.pdf

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	July 2014	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Evaluation of the Applicability of HTML5 for Mobile Applications in Resource-Constrained Edge Environments	FA8721-05-C-0003

6. AUTHOR(S)

Bryan Yan, Grace A. Lewis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2014-TN-002

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Mobile applications increasingly are being used by first responders and soldiers to support their missions. These users operate in resource-constrained, edge environments characterized by dynamic context, limited computing resources, intermittent network connectivity, and high levels of stress. In addition to efficient battery management, mobile applications operating in edge environments require efficient resource usage of onboard sensors to capture, store, and send data across networks that may be intermittent. The traditional method for building mobile applications is to use native software development kits (SDKs) on a particular mobile platform, such as Android or iOS. However, HTML5 has recently evolved to a stage where it supports many of the development features that native SDKs support. The advantages of using HTML5 not only include cross-platform development and deployment, but also that mobile edge applications would not have to be deployed on mobile devices, potentially leading to an easier distribution and testing process. This technical note presents an analysis of the feasibility of using HTML5 for developing mobile edge applications, as well as the use of bridging frameworks for filling in gaps in HTML5 development features. This note also provides a discussion of the software architecture implications of HTML5 mobile application development. The work presented in this note is the result of an independent study in Carnegie Mellon University's Master of Information Technology - Embedded Software Engineering (MSIT-ESE) program.

14. SUBJECT TERMS	15. NUMBER OF PAGES
HTML5, mobile applications, edge environment	57

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL